НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
Обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

“___”_____2020р.

Дипломний проект
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп’ютерні системи та мережі»
спеціальності 123 «Комп’ютерна інженерія»
на тему: «Криптографічний модуль для шифрування у стандарті DES»

Виконав:
студент IV курсу, групи ІО-62
Олексій ГУДЗЕНКО _____

Керівник:
Професор, д.т.н.,
Анатолій СЕРГІЄНКО _____

Консультант з нормоконтролю:
Професор, д.т.н.,
Валерій СІМОНЕНКО _____

Рецензент

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
Обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність - 123 «Комп'ютерна інженерія»
Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
"___"_____2020р.

ЗАВДАННЯ

на дипломний проект студента
Гудзенка Олексія Юрійовича
(прізвище, ім'я, по батькові)

1. Тема проекту Криптографічний модуль для шифрування у стандарті DES
керівник проекту д.т.н. професор Сергієнко А.М. ,
затверджені наказом по університету від "7" травня 2020 р. № 1081-с
2.Термін здачі студентом закінченого проекту_____.
3. Вихідні дані до проекту технічна документація.
4. Зміст пояснювальної записки
Опис та аналіз криптографічного стандарту DES, дослідження методики побудови криптографічного модуля за стандартом DES, програма криптографічного модуля для шифрування у стандарті DES.
5. Перелік графічного матеріалу (із зазначення креслеників, плакатів тощо) структурна схема, принципова схема, функціональна схема.

6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
нормоконтроль	д.т.н., проф. Сімоненко В.П.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітки
/.	*Затвердження теми роботи*	*01.09.2019*	
2.	*Вивчення та аналіз завдання*	*2.09.2019-19.02.2020*	
3.	*Розробка архітектури та загальної структури систем*	*28.02.2020-27.03.2020*	
4.	*Розробка структур окремих підсистем*	*27.03.2020-3.04.2020*	
5.	*Програмна реалізація системи*	*3.04.2020-17.04.2020*	
6.	*Оформлення пояснювальної записки*	*17.04.2020-22.05.2020*	
7.	*Захист програмного продукту*	*25.04.2020*	
8.	*Передзахист*	*26.05.2020*	
9.	*Захист*		

Студент _____ _Олексій ГУДЗЕНКО

Керівник проекту _____ _Анатолій СЕРГІЄНКО

Анотація

В бакалаврській дипломній роботі реалізовано криптографічний модуль для шифрування в стандарті DES, в основі якого полягає шифрування та дешифрування даних за алгоритмом DES.

Програма дозволяє зашифрувати введені дані та виведе результат роботи модуля шифрування.

Для виконання роботи використовується мова VHDL в середовищі ActiveHDL.

Annotation

In this work for a Bachelor's Degree, the cryptographic module for encryption in the DES standard is realized, which is based on encryption and decryption of data according to the DES algorithm.

The program allows you to encrypt the entered data and display the result of the encryption module.

The VHDL language in the ActiveHDL environment is used to perform the work.

№ з/п	Формат	Позначення	Найменування	Кількість аркушів	Примітка
1	А4		Титулка	1	
2	А4		Завдання на дипломний проект	2	
3	А4		Анотація	1	
4	А4	ІАЛЦ.467200.001 ОА	Опис альбому	1	
5	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	2	
6	А4	ІАЛЦ.467200.003 ПЗ	Пояснювальна записка	61	
7	А4	ІАЛЦ.467200.004 Д1	Додаток 1	1	
8	А4	ІАЛЦ.467200.005 Д2	Додаток 2	1	
9	А4	ІАЛЦ.467200.006 Д3	Додаток 3	1	

				ІАЛЦ.467200.001 ОА		
	ПІБ	Підп.	Дата			
Розробн.	Гудзенко О.Ю			Опис альбому	Аркуш	Аркушів
Керівн.	Сергієнко А.М.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62	
Н/контр.	Сімоненко В.П.					
Зав.каф.						

Технічне завдання до дипломної роботи

Зміст

					ІАЛЦ.467200.002 ТЗ		
Изм.	Лист	ПІБ	Підпис	Дата			
Розроб.		Гудзенко О.Ю			Розробка курсу дистанційного навчання. Технічне завдання.	Аркуш	Аркушів
Керівник		Сергієнко А.М.				1	2
Консультант						КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62	
Н/контр.		Сімоненко В.П.					
Зав. каф.							

1. Назва розробки та область її використання.

Дане технічне завдання використовується для розробки криптографічного модуля для шифрування у стандарті DES. Областю розробки являється криптографія. Використання в шифрування та дешифруванні даних.

2. Ціль розробки.

Ціллю даної роботи є аналіз криптографічних систем та розробка криптографічного модуля за стандартом DES.

3. Джерела розробки.

Джерелами для виконання роботи є науково-технічна література по криптографії, науково-технічні публікації, довідники та публікації в Інтернеті.

4. Технічні вимоги.

- шифрування-дешифрування потоку даних у конвеєрному режимі за алгоритмом DES;
- реалізація для ПЛІС.

ЗМІСТ

ВСТУП

На сьогоднішній день з розвитком мереж постає завдання надання захисту інформації. В сучасному світі є зловмисники які з метою наживи бажають отримати дані про користувачів різних сервісів, отримання таємниць фірм, тощо. Технології розвиваються і потужність обчислювальної техніки зростає, тому відкриваються нові можливості для зловмисників.

Рішенням запобігання втрати інформації є в криптографії. На даний час існує безліч алгоритмів, які мають захищати дані, але вони всі мають різну надійність. Так “національний інститут стандартів і технологій” вирішив, що потрібно прийняти єдиний стандарт для шифрування інформації. На конкурсній основі вибирався стандарт для шифрування даних.

Алгоритм DES був прийнятий за стандарт шифрування даних, він забезпечив виконання вимог щодо захисту інформації. На сьогоднішній час стандарт DES ще досі використовується в певними системами, він ще залишається актуальним для захисту інформації. Тому метою мого дипломного проекту є огляд криптографічних алгоритмів, а саме детальний розгляд алгоритму DES та його реалізація.

РОЗДІЛ 1

Теоретичні відомості про криптографію

1.1. Загальні відомості про криптографію.

З розвитком глобальної мережі Інтернету кількість систем обробки інформації значно зросла у багатьох сферах повсякденного життя. Для того, щоб забезпечити інформативність, ефективність та безпеки товарів, в продукти інтегруються вбудовані обчислювальні системи. Однак багатьом із цих невеликих систем необхідно задовольнити суворі вимоги щодо застосування щодо економічності та продуктивності. Системі потрібно керувати криптографічними алгоритмами для підтримки безпеки даних, але здійснювати це без істотного впливу на загальну продуктивність системи. Маючи ці обмеження, більшість малих мікропроцесорів не можуть забезпечити необхідну кількість криптографічних обчислень. Адже завелика кількість вбудованих систем засновані на невеликих мікропроцесорах з обмеженою обчислювальною потужністю, виконання дорогих обчислювальних криптографічних операцій на цих платформах надзвичайно складне без серйозного впливу на продуктивність. Для обробки такої обчислювальної криптографії потрібне спеціальне обладнання. Спеціально розроблені апаратні реалізації маю переваги над мікропроцесорними платформами, їх можливо оптимально спроектувати з урахуванням тимчасових і просторових особливостей більшості додатків. В даний час єдиними варіантами побудови таких апаратних чіпів для конкретного додатка є інтегральна схема (ASIC), що реалізує програму як статичну схему і програмовані логічні інтегральні схеми (ПЛІС), який дозволяє динамічне перетворення схеми додатків в

двовимірний масив загальних і реконфігурованих логічних елементів. ПЛІС дозволяють швидко створювати прототипи систем з мінімальними витратами часу і коштів на розробку. Однак ПЛІС поставляються у вигляді повного пакета з певною кількістю конфігурованої логіки, що робить використання ПЛІС для конкретного апаратного додатки більш грубим і, отже, більш дорогим у порівнянні з ASIC. ПЛІС забезпечують достатні логічні ресурси для складних реалізацій і можливість переконфігурації для поновлення реалізованих функцій безпеки в разі потреби [8], [9].

Область криптографії розділяється на два основних види криптографії: з відкритим ключем (асиметричну) та з закритим ключем (симетричну) [12]. У асиметричних алгоритмах шифрування використовуються різні ключі ради шифрування, а також розшифрування. Головне досягнення такого виду полягає в тому, що без завчасно існуючих домовленостей про заходи безпеки люди мають можливість обмінюватися секретними повідомленнями. Кожній зі сторін надається пара ключів, що складається з секретного і також відкритого ключа. Шифрування даних може здійснювати кожен, хто знає про відкритий ключ, але розшифрувати інформацію може тільки власник секретного ключа. Алгоритм шифрування та розшифрування створюється так, щоб знаючи відкритий ключ, було неможливо визначити закритий ключ.

У симетричних алгоритмах шифрування до способів шифрування та дешифрування застосовується один криптографічний ключ. Всі довірені сторони мають загальний секретний ключ, наприклад, для встановлення конфіденційного зв'язку. Даний алгоритм широко використовується в комп’ютерній техніці з системами у яких необхідно забезпечити конфіденційність інформації. Основним принципом алгоритму є умова, що сторона яка повинна приймає повідомлення знає алгоритм шифрування і ключ

до повідомлення. Без ключа зашифроване повідомлення виглядає як набір непослідовних символів та не несе будь-яку інформаційну цінність.

Симетричні криптографічні алгоритми поділяються на блокові та потокові. У потокових алгоритмах одиницею кодування є один біт. Результат шифрування не залежить від минулого вихідного потоку даних. Застосовуються в випадках коли передача інформації розпочинається та закінчується в довільні моменти часу. Блокові шифри перетворюють вхідні дані в блок вхідної інформації з фіксованою довжиною, в результаті роботи такого алгоритму отримується блок інформації такого ж обсягу як і вхідний. Застосовуються при пакетній передачі інформації та кодування файлів.

Блоковий шифр – це схема шифрування, яка розбиває звичайні текстові повідомлення, що підлягають шифруванню, на рядки, які називаються блоками, фіксованої довжини і шифрує їх по одному блоку за раз. Це відрізняється від потокових шифрів, які шифрують по одному біту за раз. Потокові шифри можна розглядати як специфічний випадок блокових шифрів, тобто коли розмір блоку становить 1 біт. Математично шифр n-бітного блоку представляє функцію виду: $E:V_n * K -> V_n$, так, що для кожного ключа K, E (P, K) є зворотнім відображенням (функція шифрування) від V_n до V_n, написаним як $E_k(P)$. Зворотне відображення - позначається функція дешифрування $D_k(P)$. $C = E_k(P)$ позначає текст шифру, який є результатом шифрування простого тексту P під ключ K.

Клод Шеннон запропонував концепції, так званої плутанини і дифузії, щоб зробити блок-шифр сильним, роблячи складним статистичний крипто-аналіз. У розповсюдженні, коли кожна цифра чистого тексту впливає на значення багатьох цифр шифрованого тексту, статистичний характер простого тексту пропадає. Плутанина призначена для того, щоб зробити зв'язок між

ключовим і зашифрованим текстом якомога більш складним.

Поняття плутанини та розповсюдження реалізуються двома класами блокових шифрів, а саме:

- Заміщення шифру;
- Перестановка шифру.

Також є третій клас комбінований, який поєднує два вище згадані шифри.

Шифри заміщення – це блокові шифри, які замінюють символи або групи символів на інші символи або групи символів.

Прості перестановочні шифри – це шифри, які просто переставляють символи в блоці.

Прості шифри заміщення та перестановки окремо не забезпечують дуже високий рівень захисту. Вони вразливі до атак методом “brutal force”. При комбінуванні цих двох шифрів, можливо отримати більш надійний шифр. Склад заміщення і перестановки називають колом. У нас може бути кілька таких кіл, щоб збільшити надійність шифру. Шеннон також припустив, що вищевказаний метод поєднання заміщення і перестановки або чергування плутанини і розповсюдження може надати новому шифрові дуже високу надійність проти зламувань. Першою практичною реалізацією цієї ідеї із застосуванням повторної плутанини та розповсюдження для шифрування був шифр Фейстеля.

Криптографія з відкритим ключем застосовується тільки в додатках, де потрібні розширені властивості безпеки асиметричного ключа. Для всіх інших потреб, таких як групове шифрування даних, симетрична криптографія є більш ефективним вибором, наприклад, з використанням стандарту шифрування даних (DES) або розширеного стандарту шифрування (AES) блоків шифрів. У багатьох випадках потрібно гібридна криптографія, що

включає в себе симетричну і асиметричну криптографію. Гібридна криптографія – це система шифрування, що поєднує переваги криптографії з відкритим ключем з продуктивністю симетричної. Симетричний ключ використовується для шифрування даних, а асиметричний для шифрування самого симетричного ключа.

Безпека симетричних і асиметричних шифрів зазвичай визначається розміром їх параметрів безпеки, зокрема довжиною ключа. При проектуванні криптосистеми ці параметри повинні бути обрані відповідно до передбачуваних обчислювальних можливостей атак. Залежно від обраної межі безпеки багато криптосистем є потенційно вразливими, здатними до атак, коли обчислювальні можливості зловмисника несподівано зростають. У реальному житті, лімітним фактором зловмисників часто є фінансові ресурси. Таким чином, з криптографічного точки зору вкрай важливо не тільки дослідити складність атаки, але і вивчити можливість зниження співвідношення витрат і продуктивності апаратного забезпечення атаки [6]. Наприклад, підвищення продуктивності атакуючої машини в 1000 разів ефективно зменшує довжину ключа симетричного шифру приблизно на 10 біт (починаючи від $1000 \approx 2^{10}$).

Основні принципи, на яких базується безпека для правильної роботи програмного забезпечення та управління комунікаціями:

- Конфіденційність: доступ до даних мають лише учасники, які беруть участь у виконанні або пересиланні повідомлень;
- Цілісність: повідомлення не повинно бути пошкоджено під час передачі і програма не повинна змінюватись перед виконанням;
- Доступність: повідомлення та програма повинні бути доступними

- Автентичність: учасник повинен бути впевнений, що повідомлення надходить від потрібної учасника при комунікації та система повинна довіряти вихідному коду програми;
- Відмово-стійкість: учасники, які беруть участь у виконанні або пересилці повідомлень, не повинні мати можливості відмови в обміні.

Крипто-аналіз сучасних криптографічних алгоритмів включає в себе масивні і паралельні обчислення, зазвичай вимагають більше 240 операцій. Багато крипто-аналітичних схеми проводять свої обчислення в незалежних операціях, що дозволяє забезпечити високу ступінь паралельності. Така паралельна функціональність може бути реалізована окремими апаратними блоками, якими можна керувати одночасно, покращуючи часову складність загальних обчислень ідеальним лінійним коефіцієнтом. На цьому етапі слід зазначити, що високі неодноразові витрати на інженерні розробки для ASIC – часто можуть витрачати більше 100000 доларів на великі проекти, що поставили більшість проектів зі створення спеціального обладнання для крипто-аналізу недоступними для комерційних або науково-дослідні установ. Однак з недавнім появою недорогих ПЛІС, в яких зберігається величезна кількість логічних ресурсів, спеціальні крипто-аналітичні машини тепер стали можливі і поза державних установ.

Існує кілька підходів до створення потужних обчислювальних кластерів для крипто-аналізу. Однак у цьому є і недолік, що полягає в тому, що успіх сильно залежить від кількості користувачів, що беруть в цьому участь. Отже, розподілені обчислення зазвичай призводять до непередбачуваного часу виконання атаки, так як доступна обчислювальна потужність змінюється через

динамічну кількість учасників. Другий природний підхід може бути заснований на використанні суперкомп'ютерів типу IBM BlueGene [Int08] або інших комерційних машин, наприклад, Cray або SGI. На жаль, суперкомп'ютери, як правило, надають складні можливості для високошвидкісного зв'язку і великих ділянок розподіленої пам'яті, які в більшості випадків не потрібні для простого крипто-аналітичної програми. На жаль, наявність цих можливостей значно збільшує вартість цих систем, що призводить до неоптимальному співвідношенню ціни і продуктивності для крипто-аналітичних додатків. З удосконаленням технології ПЛІС конфігуровані обчислення стали економічно вигідною альтернативою для деяких суперкомп'ютерних додатків.

1.2. Шифр Фейстеля.

У 1971 році Хорст Фейстель запропонував використовувати шифр, який чергує перестановку і заміщення. Він використовував концепцію шифрів продукту таким чином, щоб кінцевий результат був криптографічно-надійнішим, ніж шифри компонентів. Розроблений шифр став основою для DES.

Шифр Фейстеля – це різновид бокових шифрів з певною ітеративною структурою, який використовує один модуль для шифрування та дешифрування. Нижче на малюнку 1 описана структура шифру Фейстеля.

У шифрі Фейстела зашифрований текст розбивається на дві половини. Ці половини проходять через 'n' циклів обробки, щоб отримати текст шифру. Всі раунди мають однакову структуру. У кожному раунді застосовується кругова функція 'F' до однієї половині з допомогою підключа, як параметр

підстановки. Потім виводиться XOR функція результат на другу половину. Змінюються дві половини, а потім виконується перестановка. Колова функція також може містити деякі функції перестановки. Колова функція "F" має однакову загальну структуру протягом "n" раундів, але для неї в кожному раунді передається різний параметр у вигляді іншого підключа.

Потрібно відзначити, що шифр Фейстела використовує один і той же алгоритм як для шифрування, так і для дешифрування тільки з підключами в зворотному порядку в разі дешифрування. Для того щоб виконувати шифрування та дешифрування інформації не потрібно реалізовувати два різних алгоритми, для кожного процесу окремий. Ця властивість дуже корисна при апаратній реалізації алгоритму DES. Для процесу буде досить тільки одного контуру.

Як вже говорили, колова функція може бути односторонньою. Тоді спосіб розшифрування повідомлення цікавий. Розглянемо раунд в DES. Повідомлення розбите на 2 частини. Права половина залишається незашифрованою і поміщається в нову ліву частину. Права частина також передається в колову функцію, а результат "I" XOR в лівій частині і поміщається в якості нової правої половини. Слід зазначити, що XOR є оборотною функцією. Таким чином, під час дешифрування незашифрована ліва половина змінюється і стає правою половиною. Зашифрована права половина стає лівою половиною. Коли незашифрована права половина проходить через ту ж саму колову функцію з тим же підключем, ми отримуємо той же результат "I". Це коли XOR отримує незашифровану ліву половину. Таким чином, повідомлення розшифровується. Для декількох раундів добре, якщо підключі розташовуються у зворотному порядку.

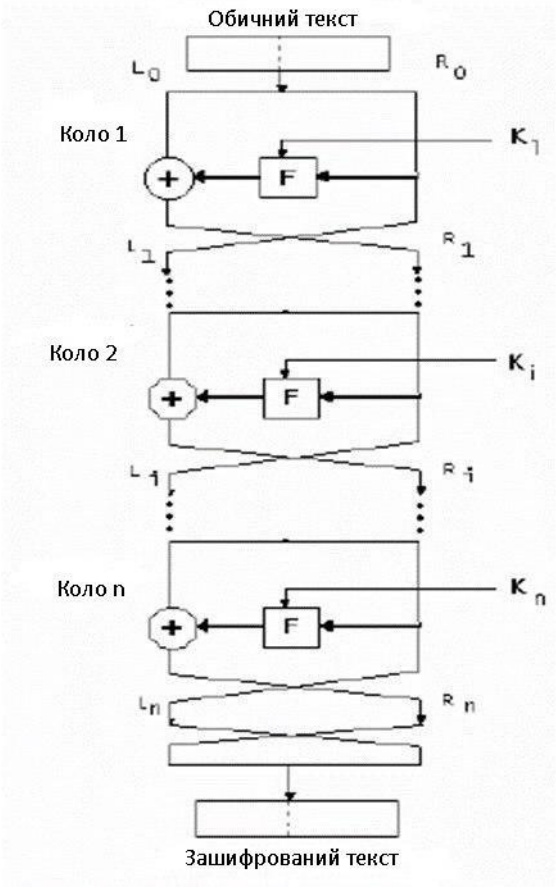


Рисунок 1.1 – алгоритм Фейстела.

Надійність шифру Фейстела залежить від ряду параметрів, які можна відрегулювати. Відповідні значення можуть бути обрані в залежності від компромісу між необхідною безпекою, швидкістю процесу і обсягом доступного простору для зберігання. Наприклад шифр Фейстеля має такі параметри:

- Розмір блоку
- Розмір ключа
- Кількість кіл
- Генерація підключів
- Характеристика колової функції

При збільшенні розмірів блоку надійність шифру підвищується, але в той же час це призводить до зменшення швидкості шифрування і до збільшення місця зберігання. Чим більший розмір ключа тим складніше зловмисникам зламати код, атаки методом “brutal force” стануть майже неможливими через велику кількість затрат на апаратуру. При збільшенні кількості кіл збільшується надійність шифру. Зазвичай це 16 кіл, як в стандарті DES. Якщо ж ускладнювати алгоритм генерації ключів, то буде велика складність при крипто-аналізі шифру і передача повідомлень при такому шифрі буде займати більшу кількість часу і апаратних можливостей. Чим більше ускладнювати колову функцію, тим надійнішим ставатиме шифр. Однак слід подбати про те, щоб обрана функція була однозначною. Структура Фейстеля також дозволяє використовувати в якості колових функцій односторонні функції. Це значно ускладнює крипто-аналіз.

Шифр Фейстела дає загальну структуру того, яким повинен бути шифр продукту, який реалізує ідею Шеннона, про повторну плутанину та дифузію. Шифр не дає певних рекомендацій для побудови алгоритму кругової функції як такої, крім того, що шифр повинен бути нелінійним, для того щоб його було важко зламати. Однією з перших реалізацій шифру Фейстела є стандарт шифрування даних (DES).

1.3. Загальні відомості про стандарт DES.

На сьогоднішній день всі фінансові операції, відео-нагляд і електронна торгівля виконуються в режимі онлайн. Всі передачі даних виконуються по таким мережам, як LAN, WAN і ATMs, які з’єднані маршрутизаторами, комутаторами, мостами та другими мережевим обладнанням. Ріст числа

віртуальних приватних мереж і рішень по забезпеченню IP-безпеки викликав підвищений попит на забезпечення передачі даних з високою надійність та продуктивністю.

Для забезпечення конфіденційності та захисту конфіденційної передачі даних від шпигунів та хакерів, уряд США рекомендує стандарт шифрування даних (DES), а також Triple DES алгоритми [1], [4], [10]. Існує декілька алгоритмів шифрування, таких як DES, Blowfish, RC4/RC5, міжнародний алгоритм шифрування даних (IDEA) та інші. Кожен з алгоритмів має свої переваги та недоліки. DES є найбільш поширеною криптосистемою відкритого стандарту, яка забезпечує гарну продуктивність.

Алгоритм блочного шифрування DES був розроблений дослідниками компанії IBM і доопрацьований державними органами. Американський національний інститут стандартів (ANSI) прийняв DES як федеральний стандарт для шифрування комерційних і конфіденційних даних. Алгоритм DES має регулярну структуру, яка піддається конвеєризації та простим маніпуляціям з даними, що дозволяє швидко працювати.

DES - це симетричний блоковий алгоритм шифрування, в якому один і той же ключ використовується як для шифрування, так і для розшифрування. DES приймає на вхід 64-бітний ключ і 64-бітний блок даних, а на виході - 64-бітові зашифровані дані. Реальний ключ становить всього 56 біт, а інші біти, тобто найменший значущий біт в кожному байті може бути використаний в якості парності. Одна і та ж базова конструкція може використовуватися як для шифрування, так і для розшифрування [11].

Алгоритм DES може шифрувати або розшифровувати будь-які дані виражені в 64-бітних словах. Вхід обробляється блоками. Тому цей алгоритм можна використовувати для захисту будь-якого типу документів, таких як:

документи, приватні дані, номера банківських рахунків, фотографій, відео, та інші типів даних.

DES є однією за найбільш широко використовуваних схем шифрування, основаної на шифрові Фейстеля. В DES дані шифруються 64-бітними блоками з використанням 56-бітного ключа. Алгоритм перетворює 64-бітний вхідні дані в 64-бітні вихідні дані за 16 раундів перетворень. Алгоритм DES призначений для шифрування та розшифрування блоків даних, які складаються із 64 біт під управлінням 56 біт [3]. Шифрований блок піддається початковій перестановці (IP), потім складному обчисленню, залежному від ключа, на кінець перестановці яка є оборотною по відношенню до початкової перестановки (IP^{-1}). Ключові залежності обчислення проводяться протягом 16 раундів і включають в себе заміщення і перестановку. DES будучи шифром Фейстеля, має той самий алгоритм шифрування та дешифрування, при умові, що підключі даються в зворотному порядку для розшифрування. Ключ зазвичай зберігається в виді 64-бітного числа, де кожні вісім бітів – це біт парності. При роботі з алгоритмом біти парності ігноруються. 56-бітний ключ використовується для створення 16 різних 48-бітних підключів, по одному для кожного раунду перетворення. 64-бітний блок, який шифрується, розбивається на дві половини. Права половина проходить через один раунд DES. Результатом є функція XOR (виключна диз'юнкція) в лівій половині, яка стає новою правою половиною. Немодифікована права половина становиться лівою половиною, так продовжується 16 раундів. Спочатку ключ передається через функцію перестановки, потім для кожного із 16 раундів створюється підключ (K_1) комбінацією із лівого кругового здвигу і перестановки. Функція перестановки однакова для кожного раунду, але через повторювані ітерації

ключових бітів генерується другий підключ. Зашифровані дані може отримати лише використовуючи той самий ключ, який використовувався для його шифрування. Зловмисники які отримали шифр та знають алгоритм, але не мають ключа для розшифрування, не можуть без значних затрат часу та застосування алгоритмів зломів отримати оригінальні дані алгоритмічно. Однак кожен, у кого є ключ, може без всяких перешкод та великих затрат часу розшифрувати шифр та отримати оригінальні дані.

В 2004 році DES позбувся статусу стандарту, але ще досі залишається популярним для систем безпеки низького рівня, а також доступний в багатьом застарілим системам. Міністерство торгівлі США вибрало розширений стандарт шифрування (AES – Advanced Encryption Standard) в якості заміни DES. Алгоритм AES – це симетричний алгоритм блокового шифрування, був розроблений в 1998 році [5]. Хоча існують побоювання з приводу безпеки DES, Triple-DES, ці стандарти шифрування ще досі вважається безпечними і рекомендується міністерством торгівлі США.

Висновок до розділу 1

В даному розділі дипломної роботи було розглянуто криптографічні стандарти, а саме зроблено увагу на типі шифру Фейстеля, до якого належить стандарт DES. Було викладено загальні відомості по криптографії та чим саме стандарт DES заслужив одне з перших місць в криптографії, чому на сьогоднішній день його менше використовують.

Серед великої кількості стандартів шифрування DES посідає одне з провідних місць, на сьогоднішній день його замінює більш новий стандарт AES, який є менш уразливим до атак методом перебору, також шифр DES удосконалюється, найбільш відомим є Triple-DES.

Для забезпечення конфіденційності власної інформації осіб стандарт DES цілком задовільнить потреби, він має високий рівень безпеки і затрати зловмисників на зламування зашифрованих даних, будуть перевищувати цінність інформації користувачів.

РОЗДІЛ 2

Аналіз та проектування алгоритму DES

2.1. Алгоритм DES.

Алгоритм DES починається з початкової перестановки (IP), шифрується шістнадцятьма "раундами", за якими слідує зворотна початкова перестановка (IP^{-1}). У кожному раунді справа наліво 32 біта блоку трансформуються з функцією з позначкою "F" і ключем, потім XOR з лівого боку 32 біта.

Початкова і кінцева перестановки в DES не виконують криптографічного функції. Спочатку вони були додані для того, щоб полегшити завантаження 64-бітних блоків в апаратне забезпечення - цей алгоритм все-таки передує 16-бітовим шинам і тепер часто опускається з реалізацій. Однак, перестановки є частиною стандарту, і тому будь-яка реалізація, що не використовує перестановки, насправді не є DES.

Нехай 64 біта вхідного блоку до ітерації складаються з 32-бітного блоку L, за яким слід 32-бітний блок R. Використовуючи цю нотацію, вхідним блоком потім буде LR. Нехай K буде блоком з 48 біт, обраним з 64-бітного ключа. Потім визначається вихід L'R' ітерації з входом LR.

Ключ для кожного раунду є підмножина вихідного 64-бітного ключа з перерахунком бітів. При кожній ітерації з 64-бітного ключа, позначеного KEY, вибирається відмінний блок K із бітів ключа. Нехай KS буде функцією, яка приймає на вхід ціле n в діапазоні від 1 до 16 і 64-бітний блок KEY. В результаті отримаємо 48-бітний блок виходу K_n, який є переставленим вибором із бітів KEY:

$$K_n = KS(n, KEY) , \quad (1)$$

де K_n – 48-бітний блок виходу, який визначається бітами в 48 різних позиціях блоку KEY;

KS – функція, яка приймає на вхід n;

n – ціле значення із діапазону [1; 16];

KEY – 64-бітний ключ.

KS називається ключовим розкладом, так як блок K, який використовується в n-й ітерації, є блоком K_n. Для n-ї ітерації лівий та правий блоки визначаються:

$$L_n = R_n - 1 , \quad (2)$$

де L_n – лівий блок для n ітерацій;

R_n – правий блок для n ітерацій.

$$R_n = L_n - 1 \oplus f(R, K_n) , \quad (3)$$

де R_n та L_n описані вище, f – функція шифрування однієї ітерації;

R – правий блок 32-бітний;

K_n – 48-бітний блок виходу;

n – значення в діапазоні від 1 до 16.

Функція шифрування "F" розширює праву сторону (R_n) до 48 біт, XOR - це біти з ключем, і розділяє отримані 48 біт на вісім 6-бітних полів. Ці поля використовуються в якості адрес на вісім 64-словне 4-бітну пам'ять, називається S-полі. Вісім 4-бітних виходів S-поле заново збираються в 32-бітове слово, а біти переставляються (P) і XOR з лівого боку (L_n) блоку. Причина розширення в тому, що один біт входу впливає на велику кількість вихідних бітів. Це показано на рисунку 2.2.

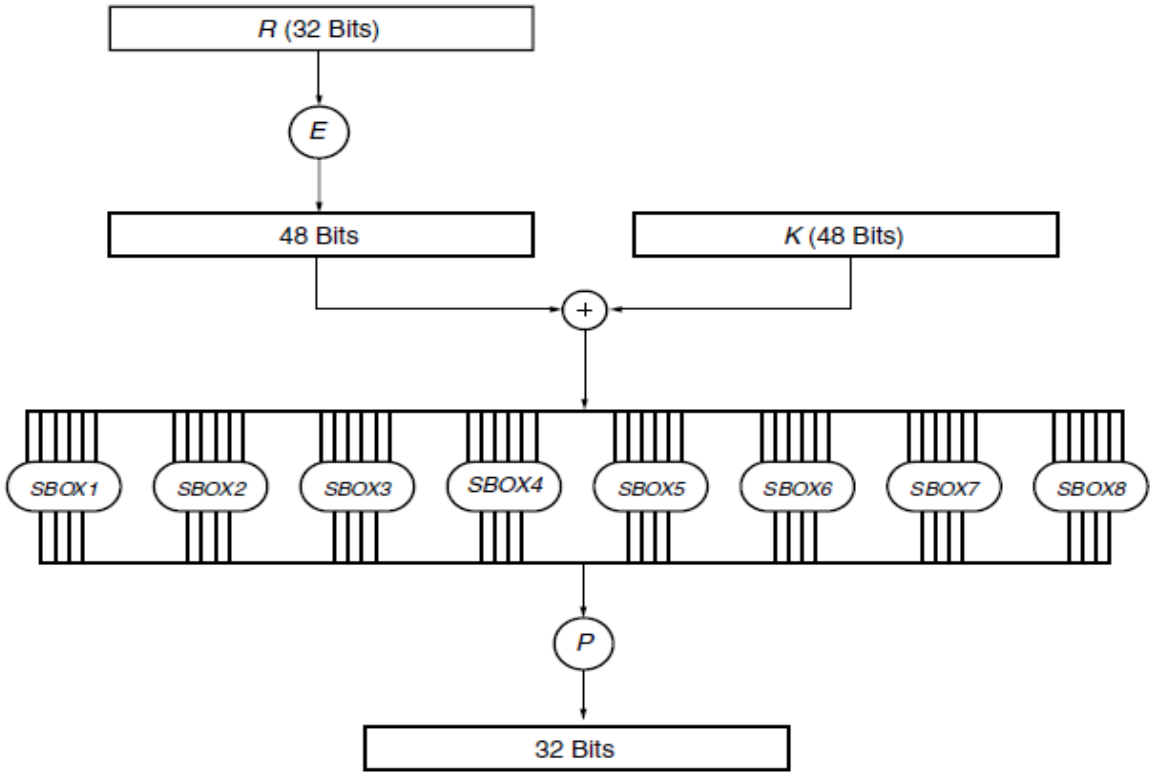


Рисунок 2.2 – обчислення функції шифрування f(R, K).

Обчислення KS показано на малюнку 2.3. Вхід KEY проходить через початковий вибір перестановки ключа, PC-1. На малюнку 2.3, C_0 представляє 32 старших біта, а D_0 - 32 молодших біта PC-1. Блоки C_n і D_n отримані з блоків

C_{n-1} і D_{n-1}, відповідно, для n = 1, 2, ..., 16, що виконується шляхом дотриманням наступного графіка зрушень окремих блоків ліворуч/праворуч, як показано на малюнку 2.3. Планування ключів під час шифрування здійснюється з використанням лівого зсуву, а під час розшифрування – правого зсуву.

Операція дешифрування аналогічна операції шифрування, але підключі K_n представлені в зворотному порядку. В апаратному забезпеченні це рівнозначно зміні напрямку зсуву і величини зсуву при генерації підключів. Перестановка IP^{-1}, що застосовується до передвихідного блоку, є зворотною по відношенню до вихідної перестановки IP, яка застосовується до входу.

Для розшифрування повідомлення необхідно застосувати той же самий алгоритм до зашифрованого блоку повідомлень. На кожній ітерації обчислення при розшифруванні використовується той же самий блок бітів ключа K, що і при кодуванні блоку.

Алгоритм DES складається з 16 однакових раундів шифрування. Кожен раунд містить значну кількість рухів бітів, яке представляє собою простий провід в апаратній реалізації, 80 2-бітних XOR і вісім пошуків в 64-словному 4-бітному S-полі. Кожен раунд використовується підмножина бітів ключів з певною перестановкою. Перестановка залежить від раунду і від того, чи буде операція шифрування або дешифрування. В першу чергу це стосується проводки, перегляду таблиць і бітових операцій. ПЛІС найкраще підходять для таких операцій і утворюють ідеальну платформу для реалізації DES.

2.2. Колова функція "F".

Для реалізації колової функції потрібно реалізувати чотири елементи:

- Таблицю розширення;
- S-поля;
- Функція перестановки;
- Генерація ключів.

Розглянемо кожен з цих елементів.

2.2.1. Таблицю розширення.

Для того, щоб один біт вхідних даних пливав більше, ніж на один біт вихідних даних, права половина даних розширюється до 48 біт. Це робить операційне розширення E, яке починається з біта 32 і циклічно повертається до початку, використовуючи всі біти по порядку, повторюючи кожен четвертий і п'ятий біт, як це було раніше.

У кожному рядку таблиці розширення повторюються перші та останні біти. Для кожного рядку, останній біт, що не повторюється, попереднього рядку та першого неповторного біту наступного рядку є бітами, які повторюються. Це створює залежність між входами у різних S-блоках у DES.

2.2.2. S-поля.

Критерії проектування для S-поля:

- S-поля повинні бути нелінійними. Якщо вони лінійні, то кожен вихідний біт може бути записаний як лінійна комбінація вхідних бітів і її легко зламати.
- Кожен рядок S-поля повинен містити всі 16 можливих вихідних комбінацій.
- Якщо два входи в S-полі відрізняються рівно на один біт, то виходи повинні відрізнятися як мінімум на два біта.
- Якщо два входи до S-полі відрізняються двома середніми бітами, виходи повинні відрізнятися принаймні двома бітами.
- Якщо два входи в S-полі відрізняються своїми першими двома бітами і однакові в двох останніх бітах, то два виходи не повинні бути однаковими.

Заміна в DES здійснюється 8 S-полями, кожен з яких приймає 6 біт на вході і виробляє 4 біта на виході. Перший і останній біти вхідного блоку разом вибирають один з 4 рядків. Середні 4 біта вибирають певний стовпець в вибраному рядку.

Вихід - це двійковий еквівалент десяткового значення в таблиці.

Наприклад, якщо вхід в S-поле S1 дорівнює 100101, то обраний рядок 11 (четвертий рядок), а стовпець 0010 (третій стовпець). Отже, на виході буде 8 (1000 в двійковому вигляді).

Можна відзначити, що кожна з рядків в S-блоці є оборотною, але в цілому S-блоки є односторонньою функцією. Це пов'язано з тим, що кожен рядок складається з перестановки чисел від 0 до 15. Тому, як тільки ми отримаємо вихід, ми не зможемо отримати назад унікальний вхід, так як ми будемо мати цей вихідний номер у всіх чотирьох рядках.

S-поля знаходяться в основі DES. В DES вони були розроблені таким чином, що навіть якщо вхідні дані різняться лише в одному біті, то після 3 раундів тексти шифрів відрізняються в 21 біті, а після 15 раундів вони різняться до 34 біт. Більш суворим критерієм є те, що будь-який вихідний біт з імовірністю 0.5 повинен змінитися при зміні будь-якого одного вхідного біта. Цей критерій, якщо він реалізований, підвищує ефективність плутанини.

2.2.3. Функція перестановки.

Функція перестановки P дає 32-бітовий вихід із 32-бітового входу, переставляючи біти блоку входу.

Критерії для перестановки P наступні:

- Чотири вихідних біта з кожного S-поля повинні впливати на шість різних S-полів в наступному раунді, і ніякі два не повинні впливати на одне і теж S-поле.
- Чотири вихідних біта з кожного S-поля на першому раунді повинні бути розподілені таким чином, щоб два з них впливали на середні біти S-поля, а решта два впливали на кінцеві біти наступного S-поля.

2.2.4. Генерація ключів.

В DES є необхідність в секретних ключах. Ключ складається з 64 бітів, з яких 56 біт генеруються випадковим чином і використовуються безпосередньо алгоритмом. Інші 8 біт, які не використовуються алгоритмом,

можуть бути використані для виявлення помилок, якщо вони інтерпретуються як біти парності. Користувач повинен мати ключ, який використовувався для шифрування даних, щоб розшифрувати їх. Використання іншого ключа спричиняє різницю шифру, який створюється для будь-якого набору входів. Криптографічна безпека даних залежить від безпеки, передбаченої для ключа, який використовується для шифрування та дешифрування даних.

Основна ідея полягає в генерації 16 48-бітових ключів з оригінальних 56 для 16 раундів так, щоб вони:

1. просто створювалися;
2. максимально відрізнялися один від одного;
3. всі ключові біти беруть участь у шифруванні всіх бітів простого тексту.

Отже, щоб зберегти секретність, нам потрібно мати ключ. В DES використовується 64-розрядний ключ, у якому кожен 8-й біт є бітом парності. Потрібно генерувати 16 48-бітних підключа. Для цього використовуються дві перестановки, які називаються PC1 і PC2. Перестановка PC-1 використовується для того, щоб зняти біти парності та генерувати 56-бітний ключ. Це також розділяє ключ на дві половини C і D:

Таким чином перша половина C буде мати біти: 57, 49, 41, 33, 25, 17, 9, 1, 58 …36; частина D аналогічно: 63… 13, 5, 28, 20, 12, 4. В кінці кожного раунду C_{i-1} і D_{i-1} підвергаються коловій зміні ліворуч, зазвичай це 1 або 2 біти, після чого вони передаються через іншу функцію перестановки, яка створює підключ для раунду. Наприклад:

1	2	3				4	5	6	7				8		
9	10	11				12	13	14	15				16		
17	18	19				20	21	22	23				24		
25	26	27				28	29	30	31				32		
33	34	35	36				37	38	39				40		
41	…														
49												…	56		
57	58	59	60				61	62	63				64		
	C							D					відкидаються		

При такій генерації можливо отримувати слабкі та напівслабкі ключі. Розділення ключових бітів навпіл робиться для того, щоб гарантувати, що всі ключові біти шифрують якомога швидше та простіше всі біти шифру. У такий спосіб, навіть якщо два ключових слова різняться в однорідному біті, отриманий шифрований текст для одного і того ж простого тексту буде сильно відрізнятися. Крім того, PC-2 вибирає 24 біти з кожної половини, в будь-якому плануванні ключів завжди є 4 біти з 56 відсутніх. Це ускладнює роботу аналітика криптографічного шифру у зворотному напрямку шляхом припущення та перевірки припущень щодо ключа. Це одна з причин вибору 48-бітного підключа, хоча можливо використовувати 56-бітний ключ.

Якщо підключі K1 - K16 рівні, то зворотній та оригінальний списки створюють ідентичні підключі: K1 = K16, K2 = K15 і так далі. Отже, функції шифрування і дешифрування збігаються. Це називається слабкими ключами.

У DES таких слабких ключів 4. Шифрування, виконане одним ключем

напівслабих пар, можна розшифрувати за допомогою іншого ключа пари. Можливі 6 таких напівслабких пар ключів. Існують "можливо слабкі ключі", які генерують лише чотири унікальні підключі, використовувані чотири рази в DES. В такому разі існує 48 "можливо слабких ключів". Ці слабкі і напівслабкі ключі були виявлені випадково і всі перераховані. Оскільки всі слабкі, напівслабкі і ймовірно слабкі ключі DES відомі, їх можна уникнути при виборі ключа.

2.3. Режими шифрування.

В даній роботі розглянемо методи, з допомогою яких можливо виконувати шифрування за алгоритмом DES для наданий даних [2]. Я пропоную виділити основні 4 типи режимів шифрування для блоків в алгоритмі DES:

- Electronic Code Book (ECB);
- Cipher Block Chaining (CBC);
- Cipher Feedback (CFB);
- Output Feedback (OFB).

Використання різних типів режимів шифрування дає різні результати надійності зашифрованого тексту. Таким чином структуру зашифрованих блоків важче проаналізувати і зламати, бо кожен блок шифру матиме власну структуру та малу імовірність, що буде повторюватися певний елемент в декількох блоках. Без використання режимів шифрування існує можливість, що при шифруванні одним ключем буде інформація, яка повторюється та є можливість, що зловмисники отримають і використають цю інформацію для розшифрування даних, тому потрібно забезпечити більшу ефективність шифру за допомогою нових методів.

2.3.1. Electronic Code Book (ECB).

Це найпростіший режим роботи, при якому відкритий текст обробляється одночасно 64-бітами і кожен блок шифрується одним і тим же ключем. Для повідомлення довшого 64-бітів, воно розбивається на 64-бітові блоки, після чого кожен з них шифрується незалежно один від одного за допомогою одного і того ж ключа.

Найбільш значуща характеристика ECB полягає в тому, що якщо один і той же 64-бітний блок з'являється кілька разів, то він завжди видає один і той же шифрований текст. Це не так безпечно для довгих повідомлень, тому що якщо повідомлення структуровано, аналітик криптографії може використовувати ці закономірності. Тому ECB підходить тільки для короткого обсягу даних. Таким чином, якщо ми хочемо таємно передати ключ DES, ми можемо використовувати ECB для шифрування ключа DES і його відправки.

2.3.2. Cipher Block Chaining (CBC).

Для того, щоб подолати недоліки ECB, використовують CBC, якщо один і той же блок повторюється, він виробляє різні блоки тексту з шифром. У цій схемі входом в алгоритм шифрування є XOR поточного блоку відкритого тексту і попереднього шифрованого текстового блоку. Один і той же ключ використовується для всіх блоків. Щоб створити перший блок зашифрованого тексту, вектором ініціалізації є XOR з першим блоком відкритого тексту перед його шифруванням. При дешифрування, вектором ініціалізації є XOR з виходом алгоритму розшифрування для отримання першого блоку відкритого тексту. Так як вхід в функцію шифрування для кожного блоку відкритого

тексту не має фіксованого відношення до блоку відкритого тексту, повторення шаблонів 64-біт призведе до появи різних текстових блоків шифру. Отже, це підходить для відправки повідомлень довше 64 біт.

2.3.3. Cipher Feedback (CFB).

CFB використовується для перетворення DES, блочного шифру в потоковий шифр. Одним з бажаних властивостей потокового шифру є те, що довжина зашифрованого тексту повинна збігатися з довжиною звичайного тексту. Це підтримується в CFB.

У CFB одиницею передачі є біти "j". У нас є 64-бітний регістр зсуву, який спочатку буде заповнений вектором ініціалізації. Він шифрується за допомогою DES з ключем K для отримання 64-бітного виходу. Найбільш значущі біти "j" в цьому висновку - це XOR по бітам "j" першого простого тексту для отримання шифрованого тексту. Ланцюг досягається в CFB тим, що цей зашифрований текст замінює останні біти "j" в 64-бітному регістрі. Цей процес триває до тих пір, поки не буде оброблено всі вхідні дані.

2.3.4. Output Feedback (OFB).

OFB схожий за структурою на CFB за винятком того, що під час ланцюжка біти "j" не беруть з тексту шифру, а передаються з виходу після етапу шифрування. Одним з переваг OFB перед CFB є те, що бітові помилки при передачі шифрованого тексту не поширюються. Але у CFB, оскільки заздалегідь створено шифрований текст використовується при розшифруванні наступних біт відкритого тексту, CFB схильний до бітових помилок.

2.4. Методи підвищення ефективності криптографічних програм.

Розвиток продуктивності глобальних мереж ставить нові завдання перед криптографічним програмним забезпеченням. Криптографічне програмне забезпечення повинне бути надійним та швидким, щоб бути актуальним. Підвищити ефективність криптографічних алгоритмів можливо створенням нового дизайну алгоритму, використовувати паралелізм, або ж можливо використовувати алгоритм незалежної апаратної підтримки.

Найбільш очевидним рішенням покращення ефективності криптографічних алгоритмів є розробка нових і швидких алгоритмів. Але це дуже важка та потрібно багато часу для розробки нового алгоритму і який буде доведеним, що захищає. Нові алгоритми вважають ненадійними. При створенні нового алгоритму використовують існуючі алгоритми, які доведені, що вони захищають дані та ці алгоритми реалізують новими методами. Також Взяті за основу алгоритми модифікують, але при модифікаціях потрібно, щоб новий алгоритм був не гіршим, ніж оригінальний. З літератури можливо вибрати будь-який алгоритм. При правильному підходу реалізації алгоритму, продуктивність має зрости. Так в алгоритмах теорії чисел, при використанні ретельних методів реалізації для представлення структури та операторів, можливе підвищення ефективності.

Другим методом підвищення ефективності шифрування криптографічних програмного забезпечення полягає в паралелізмі. Можна використовувати три типи паралелізму: паралелізм зв’язку, паралелізм пакетів та функціональний паралелізм. Паралелізм рівня зв’язку прямолінійний, але не допускається паралелізм для поліпшення пропускної здатності одного з’єднання. Паралелізм рівня пакетів пов’язаний з обробкою кожного пакету

не залежно від з’єднання. Функціональний паралелізм базується на обробці декількох частин в одному пакеті, такий спосіб можливий лише тоді, коли алгоритм шифрування пристосований для цього. Наприклад, DES з використанням Electronic Code Book дозволяє паралельно шифрувати окремі восьми-байтові блоки, таким чином паралельно можна реалізовувати при внутрішній пакетній зернистості. Однак ECB сприйнятливий до атак з простою підміною коду та підробки. Таким чином, більшість реалізацій використовують режим Cipher Block Chaining для алгоритму DES, де вихід кожного шифрування XOR переходить у наступний блок простого тексту. Хоча кожен блок DES не може бути зашифрований паралельно в режимі реалізації CBC, вони можуть бути розшифровані паралельно.

Ще одним варіантом покращення ефективності шифрування є підвищення ефективності широкого вибору криптографічного забезпечення. Наприклад: розробити класичний RISC-процесор на великому наборі реалізацій криптографічного програмного забезпечення. До класичного набору інструкцій RISC додати лише ті інструкції, які значно покращують загальну ефективність тестового набору.

2.5. Надійність алгоритму DES.

Для оцінки надійності системи враховують 3 параметри:

- Профіль зловмисника: знання про систему, наскільки великі навички, наявні засоби для зламів.
- Відмінності методів атаки.
- Цінність захищеної інформації.

Процес виявлення простого тексту або ключових бітів, або того й іншого, відомий як крипто-аналіз. Зусилля, необхідне для крипто-аналізу DES, задається порядком з числа змінних ключів, які необхідно перевірити перед тим, як найти правильний.

DES має ефективну довжину ключа 56 біт. Таким чином, кількість, кількість можливих ключів становить 256, що приблизно 7,2*1016, при такій кількості варіантів можливість типового виведення конкретного ключа вкрай малоймовірна в типових умовах загрози. Крім того, якщо ключ часто змінюється, ризик цієї події значно зменшується. Але, зі стрімким прогресом в обчислювальній техніці, кількість варіантів можливих ключів вже не здається надто великою і враховуючи простий текст та пару текстових шифрів, усі вище згадані 256 ключів можна перевірити за розумний час. Отже, DES вразливий до “brutal force”. Для покращення захисту програмного забезпечення можна використовувати алгоритми в яких покращено рівень безпеки для нашого часу, можливе використання таких алгоритмів як алгоритм Triple-DES, або ж AES.

Новий стандарт Triple-DES був запропонований в 1993 році в якості альтернативи DES. Цей алгоритм, заснований на DES, використовує три незалежних ключа (що ведуть до 168-бітного ключа) для шифрування даних. Нехай $E_k(I)$ і $D_k(I)$ представляють DES шифрування та дешифрування "I", використовуючи ключ DES – K відповідно. Кожна операція шифрування / дешифрування в Triple-DES – це складена операція шифрування та дешифрування DES, як показано на малюнку 2.4. Використовуються наступні операції:

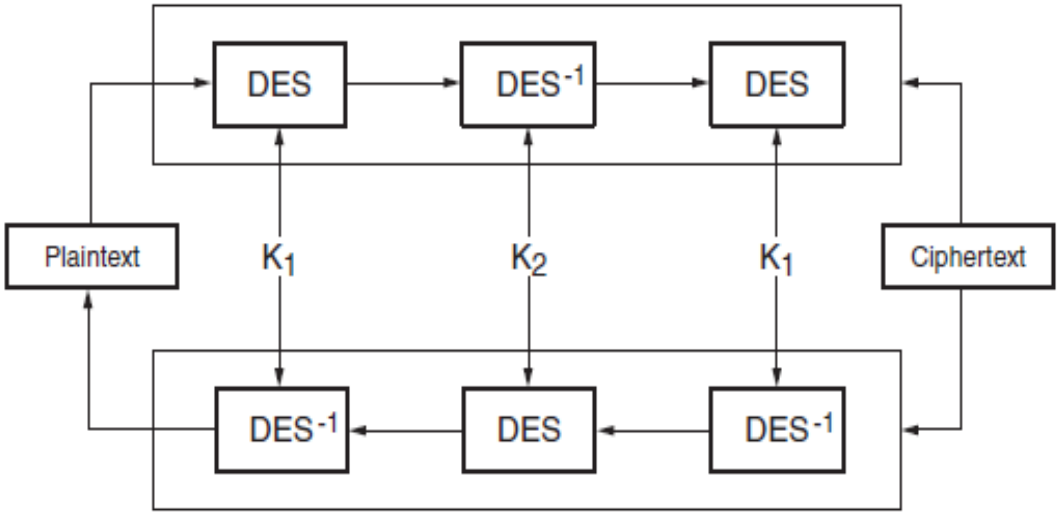


Рисунок 2.4 – Triple-DES блок-схема.

Стандарт визначає наступні варіанти ключів для пакета (K1, K2, K3):

1. Варіант ключів 1: K1, K2 і K3 - незалежні ключі.

2. Варіант ключів 2: K1 і K2 - незалежні ключі, а K3 = K1.

3. Варіант ключів 3: K1 = K2 = K3.

Варіант 1 ключів забезпечує найбільшу безпеку в той час, як варіант 2 ключів є менш захищеною. Варіант ключів 3 ідентичний одиночному DES.

DES і Triple-DES розгортаються в комунікаційному і мережевому обладнанні, де безпеку даних має першорядне значення. У мережах, здатних передавати кілька гігабіт в секунду, обладнання для шифрування не повинно ставати вузьким місцем для роботи мережі.

Найкращий вибір, який наразі є, щоб уникнути “brutal force” із застосуванням алгоритму DES - це припинити використання Single DES і перейти на Triple-DES. У Triple DES є 168-бітний ключ, який на даний момент захищений від “brutal force”. Також, використовуючи Single DES, намагатися уникати використання слабких ключів та ключів розташованих ближче до початку або до кінця простору знаходження ключів.

Основні переваги алгоритму DES:

- використання одного 56-бітового ключа;
- відносно простий алгоритм, але забезпечує високу швидкість обробки інформації;
- досить стійкий до зламування;

Слабкі місця алгоритму DES:

- ключа короткий і можливий повний перебір;
- наявність слабких та напівслабких ключів;
- можливість зламати алгоритм за допомогою лінійного та диференційного криптоаналізу.

Аналіз DES з новішим стандартом AES. AES – симетричний алгоритм блочного шифрування (розмір блоку 128 біт, ключ 128/192/256 біт), прийнятий в якості стандарту шифрування урядом США за результатами конкурсу AES [13], [14]. AES - не базується на шифрові Фейстеля, він базується на принципах нової мережі підстановок-перестановок. Має нову архітектуру “квадрат”, для якої характерно: 1) уявлення шифрованого блоку у вигляді двовимірного байтового масиву; 2) шифрування за один раунд всього блоку даних (байт-орієнтована структура); 3) виконання криптографічних перетворень, як над окремими байтами масиву, так і над його рядками і стовпцями. Це забезпечує дифузію даних одночасно в двох напрямках - по рядках і по стовпцях.

Загальні характеристики AES:

- AES зашифровує і розшифровує 128-бітові блоки даних.
- AES дозволяє використовувати три різних ключа довжиною 128, 192 або 256 біт (в залежності від довжини ключа версії шифру позначають AES-128, AES-192 або AES-256).

- Від розміру ключа залежить число раундів шифрування:
  - довжина 128 біт - 10 раундів;
  - довжина 192 біта - 12 раундів;
  - довжина 256 біт - 14 раундів.
- Всі раунди, крім останнього, ідентичні.

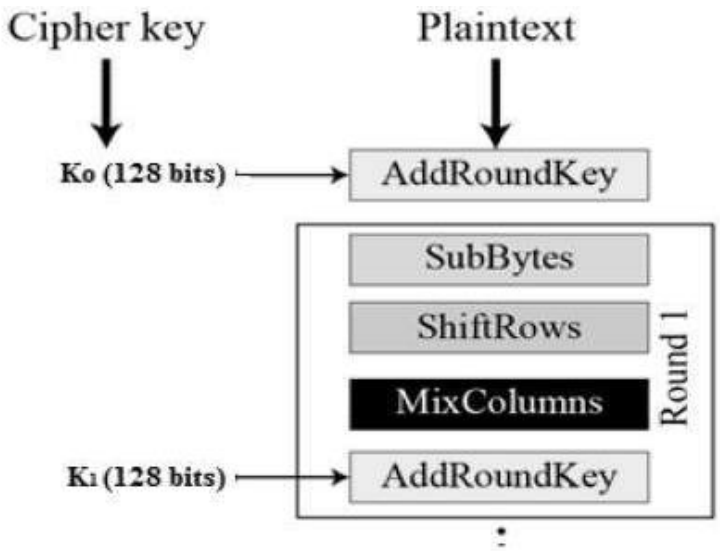


Рисунок 2.5 – схема одного проходу шифрування в AES.

Основною відмінністю між стандартами DES і AES є процес шифрування: У DES відкритий текст поділяється навпіл перед подальшим шифруванням, а у AES немає такого поділу і весь блок оброблюється разом для шифрування тексту. Також DES уступає в довжині тексту який потрібно зашифрувати та зашифрованого тексту, 64 біт, тоді як у AES – 128, 192 та 256 біт. Через невеликий розмір ключа, використовуваного в DES, він вважається менш безпечним, ніж AES, DES вважається більш уразливим для атак методом перебору. Алгоритм DES уступає в ряді параметрів перед більш новим алгоритмом AES, але він модифікують та застосовують в застарілих системах.

Висновок до розділу 2

В розділі було розглянуто алгоритм DES, його специфіка, властивості, застосування та функціонування. Алгоритм на даний час є надійним, але з його широким використанням він набув популярності та відомий для великої кількості людей, що робить його зламування більш імовірним, адже зловмисники знаючи як працює алгоритм мають змогу більш ефективніше отримувати зашифровану інформацію, витрачаючи менше зусиль. З розвитком технологій, він має недоліки перед новішими стандартами, адже доступність потужних машин збільшилась і можливість зламування методом “brutal force” зростає.

Алгоритм DES має в основі 16 кіл, які мають однакову структуру, до кожного кола генерується підключ. В залежності від режиму шифрування надійність алгоритму підвищується, виключаючи слабкі повторювані місця. Для підвищення ефективності алгоритму потрібно забезпечити паралелізм, а також обрати ефективне програмне забезпечення для криптографічних систем.

РОЗДІЛ 3

Аналіз криптографічного модуля

3.1. Характеристика ПЛІС для використання в реалізації криптографічних алгоритмів.

ПЛІС є привабливими для реалізації криптографічних алгоритмів та їх використанні в захисті даних по ряду причин, які будуть розглянуті нижче.

ПЛІС надають декілька ключових функцій для їх використання в цілях захисту даних. Наприклад: використання технології SRAM надає можливість динамічно змінювати функціональність системи, для реагування на атаки. Також можна оновити деякі нові апаратні криптографічні ядра шляхом віддаленої переконфігурації, навіть якщо система вже використовується протягом декількох років. Це допомагає підтримувати рівень безпеки системи в той час, коли методи крипто-аналізу постійно удосконалюються.

Вище були приведені основні принципи безпеки для правильної роботи програмного забезпечення та управління комунікаціями:

- конфіденційність;
- цілісність;
- доступність;
- автентичність;
- відмово-стійкість.

Виконанні всіх цих пунктів в системі є інтелектуально та фінансово дорогим. Зловмисники для порушення одного з пунктів використовують два методи: витяг секретної інформації (або ключів, які використовуються для

шифрування даних); або збій в роботі системи.

Вимоги до безпеки криптографічних модулів випливають з наступних високо-рівневих функціональних цілей в області безпеки:

- Використовувати і правильно реалізовувати затверджені функції безпеки для захисту конфіденційної інформації.
- Для захисту криптографічного модуля від несанкціонованої експлуатації або використання.
- Для запобігання несанкціонованого розкриття вмісту криптографічного модуля, включаючи прості текстові криптографічні ключі та інші критичні параметри безпеки (CSP).
- Для запобігання несанкціонованій і непоміченою модифікації криптографічного модуля і криптографічних алгоритмів, в тому числі несанкціонованої модифікації, підстановки, вставки і видалення криптографічних ключів і CSP.
- Для забезпечення індикації робочого стану криптографічного модуля.
- Для забезпечення належного функціонування криптографічного модуля при роботі в затвердженому режимі.
- Виявлення помилок в роботі криптографічного модуля і запобігання компрометації конфіденційних даних і CSP, що виникають в результаті цих помилок.

Хоча вимоги до безпеки, зазначені в стандарті FIPS 140-2, призначені для підтримки безпеки, що забезпечується криптографічним модулем, відповідність цього стандарту необхідно, але недостатньо для забезпечення безпеки конкретного модуля. Оператор криптографічного модуля несе

відповідальність за забезпечення того, щоб безпеку, що надається модулем, була достатньою і прийнятною для власника інформації, що захищається, а також за визнання і прийняття будь-якого залишкового ризику.

Аналогічним чином, використання затвердженого криптографічного модуля в комп'ютерній або телекомунікаційної системи не є достатнім для забезпечення безпеки всієї системи в цілому. Загальний рівень безпеки криптографічного модуля повинен забезпечувати рівень безпеки, що відповідає вимогам безпеки, що пред'являються до застосування і навколишньому середовищу, в якій повинен використовуватися модуль, а також до забезпечуваним модулем службам безпеки.

Стандарт визначає чотири якісних рівня безпеки - від 1-го до 4-го. Ці чотири зростаючих рівня безпеки дозволяють створювати економічно ефективні рішення, які підходять для різних ступенів чутливості даних і різних прикладних середовищ.

Рівень безпеки 1 вимагає використання тільки затвердженого алгоритму або функції безпеки. Він дозволяє виконувати програмні та програмно-апаратні компоненти криптографічного модуля на обчислювальної системі загального призначення з використанням на минулому атестацію операційної системи. Для криптографічного модуля рівня безпеки 1 не потрібно ніяких спеціальних фізичних механізмів захисту, крім базового вимоги для компоновок виробничого класу.

Рівень безпеки 2 підвищує фізичну безпеку модуля, додаючи вимоги до доказів злому і завіренні оператора на основній ролі. Починаючи з рівня 2 слід використовувати надійну операційну систему з затвердженим ступенем безпеки.

У рівні безпеки 3 механізми захисту повинні виявляти та реагувати на спроби фізичного доступу, несанкціонованого використання чи модифікації криптографічного модуля. Схема виявлення несанкціонованих дій повинна "обнуляти" всі CSP прямого тексту при виявленні фізичного вторгнення. На цьому рівні безпеки завірення на основі ролей змінюється на рівні ідентифікації. На рівні 3 введення / вивідок CSP простого тексту слід виконувати за допомогою портів, які фізично відокремлені від інших портів даних.

Рівень безпеки 4 є найвищим рівнем безпеки. Модуль повинен бути укладений в повний захисний конверт. Проникнення в модуль повинен мати дуже високу ймовірність виявлення, і модуль повинен виявляти коливання умов навколишнього середовища, а саме напруга і температурні антени, які виходять за межі нормального робочого діапазону.

Управління ключами також є однією з основних задач для вбудованих систем. Ця загроза не специфічна для ПЛІС, ASIC також уразливі. У криптографічних пристроях необхідно зберігати деякі симетричні і асиметричні ключі для виконання шифрування або розшифрування. Під час використання системи ці криптографічні ключі повинні бути змінені або згенеровані випадковим чином. Тому такі типи пристроїв вимагають безпечного управління ключами.

Для розшифрування інформації зловмиснику потрібен ключ шифрування. Одним із способів отримання ключів шифрування є прослуховування бічних каналів. Такий вид атаки називається атакою по бічних каналах. Іншим способом отримання ключів шифрування є тимчасовий аналіз або атака за часом. Тимчасова реакція системи на витік інформації

дозволяє зловмисникові отримати ключ шифрування або іншої важливої інформації, такої як пароль.

Механізми шифрування в бітовому потоці доступні для більшості ПЛІС, що працюють в режимі реального часу. Секретний ключ шифрування зберігається всередині програмованого пристрою, в той час як для енергонезалежних ПЛІС для підтримки значення ключа використовується зовнішній акумулятор. Таким чином, пристрій приймає зашифрований потік бітів і використовує його спеціальне дешифрування для отримання нешифрованих даних. Зловмисники не можуть розшифрувати потік бітів без секретного ключа. Завдяки цій можливості зловмисники повинні виявити секретний ключ, необхідний їм для відновлення бітового потоку даних, за допомогою атаки.

Потенціальні переваги ПЛІС для криптографічних систем:

- Алгоритм швидкості. Цей термін відноситься до перемикання криптографічних алгоритмів під час роботи цільової програми. Хоча гнучкість алгоритму дорога з традиційними програмними засобами, ПЛІС можна перепрограмувати на ходу.
- Алгоритм загрузки. Модернізація польових пристроїв можлива з використанням нового алгоритму шифрування. Пристрої шифрування на ПЛІС можуть завантажувати новий код налаштувань.
- Архітектурна ефективність. У певних випадках апаратна архітектура може бути набагато ефективнішою, якщо вона розроблена для певного набору параметрів. Одним із прикладів параметра для криптографічних алгоритмів є ключ. ПЛІС дозволяє

апаратно оптимізувати за допомогою певного набору параметрів, в залежності від типу ПЛІС, додаток може бути повність або часткового покращено.

- Ресурсна ефективність. Більшість протоколів безпеки - це гібридні протоколи, яким потрібно кілька алгоритмів. Оскільки вони не використовуються одночасно, то ПЛІС можна використовувати для реконфігурації часу виконання.
- Алгоритм модифікації. Є програми, які потребують модифікації стандартизованих криптографічних алгоритмів.
- Пропускна здатність. Мікропроцесори загального призначення не оптимізовані для швидкого виконання шифрування. Хоча ПЛІС повільніше, ніж реалізація ASIC, але може запускатись набагато швидше, ніж реалізація програмного забезпечення (як у процесорі).
- Економічна ефективність. Є пара факторів, які слід враховувати при аналізі економічної ефективності ПЛІС: вартість розробки та ціна одиниці продукції. Вартість розробки програми на ПЛІС для певного алгоритму значно нижча, ніж для реалізації ASIC. Ціни на одиницю продукту ASIC не є високими порівняно з витратами на розробку. Однак для застосувань з великим обсягом, наприклад більше мільйону схем, ASIC, як правило, є найбільш економічним вибором.

3.2. Особливості безпеки криптографічного модуля.

Вимоги до безпеки, визначені в стандарті FIPS 140-2 [7], охоплюють 10 областей, пов'язаних з розробкою та реалізацією криптографічного модуля:

- специфікації криптографічного модуля;
- порти та інтерфейси модулів;
- ролі, послуги та автентифікація;
- моделі кінцевих станів;
- фізична безпека;
- операційне середовище;
- управління криптографічними ключами;
- електромагнітні перешкоди /електромагнітна сумісність;
- само-тестування;
- забезпечення дизайну.

Остання область, що стосується пом’якшення інших атак, ще не перевірена, але постачальник зобов'язаний документувати реалізовані елементи управління. Хоча більшість із цих областей пов'язані з використовуваною технологією, деякі стосуються лише методології проектування.

Специфікація криптографічного модуля стосується поєднання апаратних, програмних та вшитого програмного забезпечення, що використовуються для реалізації криптографічних функцій та протоколів у межах визначеної криптографічної межі. Криптографічний модуль повинен реалізувати принаймні одну затверджену функцію захисту, що використовується в затвердженому режимі роботи. Криптографічна межа повинна складатися з чітко визначеної області, яка встановлює фізичні межі

криптографічного модуля. Якщо криптографічний модуль складається з програмного забезпечення або компонентів мікро-програмного забезпечення, криптографічна область повинна містити процесор та інші апаратні компоненти, які зберігають і захищають компоненти програмного забезпечення та мікро-програмного забезпечення. Компоненти програмного забезпечення, програмного забезпечення та мікро-програмного забезпечення криптографічного модуля можуть бути виключені з вимог безпеки, якщо буде показано, що ці компоненти не впливають на безпеку модуля.

Порти і інтерфейси криптографічних модулів - два ієрархічних рівня портів введення / виводу визначаються стандартом: фізичні порти і логічні інтерфейси. Криптографічний модуль повинен мати щонайменше чотири логічні інтерфейси: ввід даних, вивід даних, керуючий вхід і вихідний стан, які можуть спільно використовувати один фізичний порт. Ввід та вивід даних використовуються для передачі даних у простому тексті та зашифрованому тексті. Починаючи з рівня безпеки 3, порт вводу / виводу для простого тексту CSP повинен бути фізично відділений від інших портів даних. Всі дані, що виводяться через інтерфейс виведення даних, повинні бути попереджені при наявності помилки і під час само-тестування. Інтерфейс вводу управління служить для надання команд і сигналів, а також для управління даними. Інтерфейс виводу стану дозволяє конструкції надсилати сигнали, індикатори та інформацію про стан.

Фізична безпека - криптографічний модуль захищений фізичною апаратурою. Механізми обмеження несанкціонованого фізичного доступу до вмісту модуля і запобігання несанкціонованого використання або модифікації модуля (в тому числі заміна всього модуля) при установці. Все апаратне і програмне забезпечення, вбудоване ПО і компоненти даних, що знаходяться в

межах криптографічного кордону, повинні бути захищені. Фізико-технічні вимоги до безпеки вказуються для трьох певних фізичних даних криптографічного модуля:

У одно-кристальних криптографічних модулях в якості самостійного пристрою використовується мікросхема з однієї інтегральної схемою (IC). Прикладами одно-кристальних криптографічних модулів є смарт-карти з однієї інтегральної мікросхемою. Хоча одно-кристальні криптографічні модулі найбільш уразливі для атак по бічних каналах, вони зазвичай використовуються у ворожому середовищі. Модулі з одним мікропроцесором на базі ПЛІС досі не використовувалися, оскільки в них використовується незалежна технологія.

- Багато-кристальні вбудовані криптографічні модулі - це фізичні пристрої, в яких два або більше мікросхем з'єднані між собою. По можливості вони повинні бути вбудовані в непрозорий корпус. Прикладами багато-кристальні вбудованих криптографічних модулів є адаптери і плати розширення.
- Багато-кристальні автономні криптографічні модулі є фізичними втіленням, в яких дві або більше мікросхеми з'єднані між собою, а весь корпус фізично захищений. Приклади багато-кристальних автономних криптографічних модулів включають маршрутизатори шифрування або захищені радіостанції.
- Операційне середовище криптографічного модуля - це управління програмним забезпеченням, прошивкою і апаратними компонентами, необхідними для роботи модуля. Робоче оточення може бути як немодифікованим, так і модифікованим. Операційна

система є важливим компонентом операційного середовища криптографічного модуля.

Управління криптографічними ключами - вимоги безпеки для управління криптографічними ключами охоплюють весь життєвий цикл криптографічних ключів, криптографічних компонентів ключа і CSP, використовуваних криптографічних модулем. Управління ключами включає в себе генерацію випадкових чисел і ключів, установку ключів, атрибуцію ключів, ввід / вивід ключів, зберігання ключів і “обнулення” ключів. Секретні ключі, приватні ключі і всередині криптографічного модуля CSP повинні бути захищені від несанкціонованого розкриття, модифікації і підміни. Усередині криптографічного модуля відкриті ключі повинні бути захищені від несанкціонованого зміни і підміни. Криптографічний модуль може використовувати генератори випадкових чисел для внутрішньої генерації криптографічних ключів та інших CSP. Є два основні класи генераторів: детермінований та недетермінований. Детермінований складається з алгоритму, який генерує послідовність бітів з початкового значення, званого насінням. Недетермінований дає вихід, який залежить від якогось непередбачуваного фізичного джерела, що знаходиться поза контролем людини. Не існує схвалених FISP недетермінованих генераторів випадкових чисел. Якщо значення генерації проміжних ключів надсилаються поза криптографічного модуля, то значення повинні надсилатися або в зашифрованому вигляді, або в рамках процедур поділу знань. Створення ключів може здійснюватися автоматизованими методами (наприклад, з використанням алгоритму відкритого ключа), ручними методами (з використанням ручного пристрою завантаження ключів) або комбінацією автоматизованого і ручних методів. Криптографічні ключі, що зберігаються

всередині криптографічного модуля, повинні зберігатися або у відкритому тексті, або в зашифрованому вигляді. Секретні і закриті ключі у відкритому тексті не повинні бути доступні неавторизованих операторам ззовні криптографічного модуля. Криптографічний модуль повинен асоціювати криптографічний ключ (секретний, приватний або загальнодоступний), що зберігається всередині модуля, з правильним об'єктом (наприклад, особою, групою чи процесом), якому призначений ключ. Криптографічний модуль повинен надавати метод для “обнулення” всіх секретних і закритих криптографічних ключів і CSP всередині модуля. “Обнулення” зашифрованих криптографічних ключів і CSP або ключів, фізично або логічно захищених іншим чином всередині додаткового вбудованого підтвердженого модуля, що відповідає вимогам даного стандарту, не потрібно.

Криптографічний модуль повинен виконувати самостійне тестування живлення та умовні само-тести, щоб забезпечити коректне функціонування модуля. Само-тестування повинне виконуватися під час ввімкнення криптографічного модуля до живлення. Умовні само-тести повинні виконуватися, коли викликається відповідна функція безпеки або операція само-виклику. Якщо криптографічний модуль не проходить само-тестувати, модуль повинен ввести стан помилки та вивести індикатор помилки через інтерфейс виводу стану. Криптографічний модуль не повинен виконувати жодних криптографічних операцій, перебуваючи в стані помилки. Усі вихідні дані слід гальмувати, коли існує стан помилок. Криптографічний модуль повинен виконувати наступні тести при ввімкненні: тест криптографічного алгоритму, тест на цілісність програмного забезпечення та мікро-програмного забезпечення та перевірку критичних функцій. Він також може виконувати наступні умовні само-тесту: тест на парність, тест на навантаження

програмного забезпечення та вшитого програмного забезпечення, тест на введення ключа вручну, безперервне тестування генератора випадкових чисел та обхідне тестування.

Політика безпеки криптографічного модуля повинна складатися з специфікації правил безпеки, відповідно до яких повинен функціонувати криптографічний модуль, включаючи правила безпеки, що випливають з вимог стандарту і додаткових правил безпеки, встановлених постачальником. Існує три основні причини необхідності політики безпеки:

- Це потрібно для перевірки FIPS 140-2.
- Це дозволяє людям та організаціям визначати, чи відповідає криптографічний модуль, який реалізується, задовольняє заявленій політиці безпеки.
- Це описує можливості, захист та права доступу, що надаються криптографічним модулем, дозволяє особам та організаціям визначати, чи відповідає він їхнім вимогам безпеки.

Важливо динамічно адаптувати продуктивність та забезпечувати безпеку модуля. З точки зору безпеки, це дозволяє глобальній системі вести себе як рухома ціль, тоді як, з точки зору продуктивності, вона дозволяє динамічно використовувати різні пропускні здатності залежно від реальних вимог програми.

Висновок до розділу 3

В даному розділі було розглянуто специфіку криптографічних модулів на ПЛІС. ПЛІС має ряд корисних властивостей, щоб використовуватися для криптографічних систем. Системи на основі ПЛІС дешевші, аніж на процесорах, також елементна база ПЛІС може бути підібраною для конкретної системи, що дає перевагу в ефективності обчислень. ПЛІС є надійним апаратним засобом для використання криптографічних систем. Також системи розроблені на ПЛІС не вийдуть з використання так швидко, як на процесорах, адже є можливість перенесення програмного забезпечення на більш нові апаратні реалізації.

РОЗДІЛ 4

Розробка криптографічного модуля DES

4.1. Вибір засобів проектування криптографічного модуля.

Для реалізації криптографічного модуля на даний момент є багато засобів, які успішно використовуються, але потрібно обрати який буде найбільш підходити для моєї розробки. Найперше потрібно вибрати мову програмування, яка буде гарно підходити для реалізації поставленої задачі. Для криптографічних завдань використовуються певний діапазон програмного забезпечення. Наразі широко поширене використання криптографічних систем на ПЛІС. Такі системи є високопродуктивними та мають переваги в конкретному налаштуванні на поставлену задачу, таким чином можуть бути додані потрібні елементи, а також весь спектр елементів, як на процесорах, відсутній, що робить ПЛІС значно привабливіше по ціні, а також ефективності.

Для розробки на ПЛІС використовують мови програмування HDL. Для розробки я вибрав мову VHDL. Мова використовується для опису апаратних рішень інтегральних схем, що дає переваги в ефективності роботи систем які нею описані.

VHDL має ряд унікальних властивостей, щоб використовуватися при розробці систем:

- Простота в виявленні помилок, мова має чіткі стандарти опису, що дозволяє легко перевіряти правильність розробок;
- Можливість опису на багатьох рівнях: структурний, логічний, алгоритмічний;

- Довго-тривалість використання рішень розроблених на мові VHDL. Проект може бути перенесений на більш нову елементу базу без великих затрат.

Також зазначимо, що структура компонентів при реалізації буде мати вигляд типу:

1. Спочатку ми прописуємо бібліотеки, які будемо використовувати та зазначаємо які саме пакети будуть використовуватися, наприклад:

   library ieee;

   use ieee.std_logic_1164.all;

2. Потрібно об’явити об’єкт з яким будемо працювати, наприклад:

   entity “ ім’я об’єкту ” is

3. Також потрібно визначити порти для вводу інформації, виводу оброблених даних, де вказується тип даних, також можливо вказати діапазон для портів, наведемо приклад:

   port (“ім’я порта” : “вказуємо чи приймає дані, чи передаємо (in, out)” (“також за необхідністю вказуємо діапазон для порту (1 to 64 / 64 downto 1)”);

   Закінчується опис інтерфейсу наступним чином:

   end “ ім’я об’єкту ”;

4. Опис об’єкту починається в архітектурі зі слова architecture, тут описується реалізація:

   architecture “ідентифікатор” of “ім’я об’єкту” is

   “оголошення в блоці”

   begin

   “тіло архітектури”

   end “ ідентифікатор ”;

4.2. Визначення компонентів для криптографічного модуля DES.

Для розробки модуля потрібно визначити пріоритетні компоненти алгоритму для реалізації. Початок алгоритм DES будемо реалізовувати з початкової перестановки.

4.2.1. Початкова перестановка.

В початковій перестановці вводиться простий текст, який шифрується, в якій потрібно провести заміну бітів вхідних даних. На даному етапі не використовуються певні складні логічні операції. Фактично в цьому компоненті ми будемо вводити дані з певними замінами бітів.

Код для цього об’єкту для детальнішого розгляду:

```
library ieee;
use ieee.std_logic_1164.all;
entity IP is
port (d_in : in std_logic_vector(1 to 64);
            per_d_l : out std_logic_vector(1 to 32);
            per_d_r : out std_logic_vector(1 to 32) );
end IP;
architecture behavior of IP is
begin
    per_d_l(1) <= d_in(58);
    per_d_l(2) <= d_in(50);
    per_d_l(3) <= d_in(42);
    per_d_l(4) <= d_in(34);
    per_d_l(5) <= d_in(26);
    per_d_l(6) <= d_in(18);
```

```
per_d_l(7) <= d_in(10);
per_d_l(8) <= d_in(2);
……
per_d_l(31) <= d_in(16);
per_d_l(32) <= d_in(8);
per_d_r(1) <= d_in(57);
per_d_r(2) <= d_in(49);
per_d_r(3) <= d_in(41);
……
per_d_r(31) <= d_in(15);
per_d_r(32) <= d_in(7);
end;
```

Як видно з коду складних операцій в цьому об’єкті немає, тільки введення даних з початковою перестановкою бітів. Надалі буде приводитись код тільки з частини архітектури.

4.2.2. Генерація ключів.

Наступним потрібно згенерувати ключі, ця частина включатиме дві перестановки, які називаються PC1 і PC2. PC1, і PC2 - це біти, що перебувають у складі компонентів. PC1 відкидає 8 біт від ключа, який спочатку становить 64 біта. PC2 також відкидає деякі біти, щоб зменшити кількість біт з 56 до 48.

Це було реалізовано наступним чином:

```
begin
    PC_1: PC1 port map ( key => key, c0 => c0, d0 => d0 );
    c1 <= To_StdLogicVector(to_bitvector(c0) rol 1);
    d1 <= To_StdLogicVector(to_bitvector(d0) rol 1);
    c2 <= To_StdLogicVector(to_bitvector(c1) rol 1);
    …….
    d16 <= To_StdLogicVector(to_bitvector(d15) rol 1);
```

```
    PC2_1: PC2 port map ( c => c1, d => d1, k => k1 );
    PC2_2: PC2 port map ( c=>c2, d => d2, k => k2 );
    …….
    PC2_16: PC16 port map ( c=>c16, d=>d16, k=>k16 );
end;
```

На цьому фрагменті коду показано генерування підключів для 16 кіл алгоритму за допомогою функцій PC1 та PC2. Функція PC1 має досить просту структуру, вище було описано як працює дана функція:

```
architecture behavior of pc1 is
    signal Sig_pc1 : std_logic_vector(1 to 56);
begin
    Sig_pc1(1) <= key(57);
    Sig_pc1(2) <= key(49);
    Sig_pc1(3) <= key(41);
    …….
    Sig_pc1(55) <= key(12);
    Sig_pc1(56) <= key(4);
    c0<= Sig_pc1(1 to 28);
    d0<= Sig_pc1(29 to 56);
end behavior;
```

Функція PC2 відкидає деякі біти, щоб зменшити кількість бітів з 56 до 48:

```
 architecture behavior of PC2 is
     signal Sig_pc2 : std_logic_vector(1 to 56);
 begin
     Sig_pc2(1 to 28) <= c;
     Sig_pc2(29 to 56) <= d;
     sub_key(1) <= Sig_pc2(14);
     sub_key(2) <= Sig_pc2(17);
    …….
```

```
        sub_key(47) <= Sig_pc2(29);
        sub_key(48) <= Sig_pc2(32);
    end behavior;
```

4.2.3. Колова функція.

Колова функція повторюється 16 разів та має поєднувати в собі компоненти, які складають її конструкцію. Компоненти виконують різний функціонал, такий як: розширення кількості бітів з 32 до 48 бітів; виконання операції XOR для підключів та вхід до колової функції; 8 S-полів, які є оглядовими таблицями; перестановка бітів; виконання операції XOR для результату перестановки лівої частини шифрованого тексту.

Наведемо приклади компонентів колової функції. Першим наведемо приклад для S-поля:

```
architecture behaviour of S1 is
begin
process(b)
begin
case b is
    when "000000" => So <= To_StdLogicVector(Bit_Vector'(x"e"));
    when "000010" => So<=To_StdLogicVector(Bit_Vector'(x"4"));
    when "000100" => So <= To_StdLogicVector(Bit_Vector'(x"d"));
    when "000110" => So <= To_StdLogicVector(Bit_Vector'(x"1"));
    when "001000" => So <= To_StdLogicVector(Bit_Vector'(x"2"));
    ……..
    when "111101" => So <= To_StdLogicVector(Bit_Vector'(x"6"));
    when others => So <= To_StdLogicVector(Bit_Vector'(x"d"));
end case;
end process;
end;
```

Розширення бітів відбувається без значних логічних ресурсів:

```
architecture behavior of expansion is
begin
    exp(1) <= b_r_in(32);
    exp(2) <= b_r_in(1);
    exp(3) <= b_r_in(2);
    ……..
    exp(47) <= b_r_in(32);
    exp(48) <= b_r_in(1);
end behavior;
```

Виконання операції XOR для підключів та вхід до колової функції, призначення для S-полів діапазонів вхідних даних:

```
architecture behavior of funcXORin is
is signal x: std_logic_vector( 1 to 48);
begin
   x <= subkey xor a;
   S_b1 <= x(1 to 6);
   S_b2 <= x(7 to 12);
   S_b3 <= x(13 to 18);
    ……..
   S_b8 <= x(43 to 48);
end behavior;
```

4.2.4. Фінальна перестановка.

Фінальна перестановка є оберненою до початкової перестановки і не витрачає багато ресурсів:

```
architecture behaviour of FP is
begin
    f_d(1)<=d_r(8);
    f_d(2)<=d_l(8);
    f_d(3)<=d_r(16);
    f_d(4)<=d_l(16);
    f_d(5)<=d_r(24);
    …….
    f_d(63)<=d_l(25);
    f_d(64)<=d_r(25);
end;
```

4.3. Реалізація.

Реалізація криптографічного модуля для шифрування в стандарті DES є простою, для графічного інтерфейсу, показується інформація про простий текст, ключ шифру, час та перевірка відповідності вхідних даних зашифрованим даним. Таким чином можливо наглядно спостерігати за роботою алгоритму.

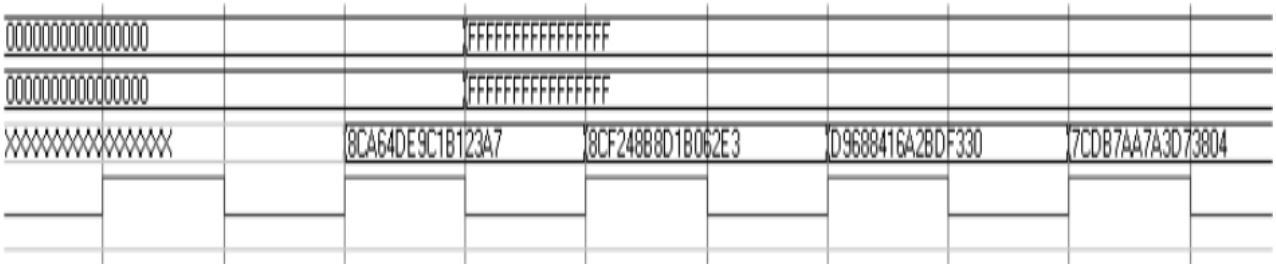


Рисунок 4.1 – Моделювання.

Висновок до розділу 4

Було розглянуто засоби проектування криптографічного модуля. Враховуючи вимоги до програми дипломної роботи було обрано мову VHDL, яка має ряд корисних властивостей. Обрана мова має чітку структуру, що дозволяє ефективно вдосконалювати код та знаходити помилки.

В даному розділі було розглянуто методи, які потрібно реалізувати для алгоритму DES та їх компоненти, структура проекту.

Програма завдяки обраній мові VHDL має зрозумілий код та без великих затрат в часі для розбору роботи може використовуватися для шифрування заданих даних.

Висновки

В даній дипломній роботі розглянуто криптографічні стандарти для розробки криптографічного модуля шифрування в стандарті DES.

Було проаналізовано криптографічні стандарти на сьогоднішній час, їх актуальність використання. Серед розглянутих варіантів стандарт DES є ще досі актуальним в використанні, хоча появились більш нові стандарти. Висока ефективність в захисті алгоритму DES, забезпечує широке використання його в світі. Порівнюючи з іншими стандартами DES є швидким та надійним, хоча в порівнянні з більш новим алгоритмом поступається в цих параметрах. Наразі алгоритм DES має модифіковані версії, які з розвитком технологій не поступаються новому стандарту AES.

Після аналізу наявних стандартів, було описано алгоритм DES. Детально описано елементи алгоритму. Розглянуто методи підвищення ефективності криптографічних програм.

Для розробки криптографічного модуля потрібно обрати апаратне забезпечення на яке буду реалізовуватися програма, було визначено, що актуальним буде розробка для ПЛІС, адже вони мають низку переваг в використанні. Також було визначено яким на сьогоднішній день повинен бути криптографічний модуль.

В результаті було розроблено криптографічний модуль для шифрування в системі DES на мові VHDL. Програма завдяки властивостям обраної мови є зрозумілою для читання, а також для перевірки роботи. Було реалізовано вивід потрібних даних для показу необхідних параметрів в перевірці роботи модуля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. FIPS, “Data Encryption Standard”, Federal Information Processing Standards Publication 46-3, October 1999.

2. Schneier, B., *Applied Cryptography*, John Wiley and Sons, 1996.

3. S. Trimberger, et. al. “A 12Gbps DES Encryptor/Decryptor Core in an FPGA.” Cryptographic Hardware and Embedded Systems - CHES 2000, Springer Verlag, 2000.

4. ANSI, “Triple Data Encryption Algorithm Modes of Operation”, American National Standards Institute X9.52-1998, American Bankers Association, Washington DC, July 29, 1998.

5. Elbirt, A.J., Yip, W., Chetwynd, B., Paar, C.: An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. IEEE Trans, 2001.

6. Dandalis, A., Prasanna, V.K.: An adaptive cryptographic engine for Internet protocol security architectures. ACM Trans. Des. Autom. Electron, 2004.

7. National Institute of Standards Technology: Security requirements for cryptographic modules (FIPS pub 140-2). In: Federal Information Processing Standards Publication. National Institute of Standards and Technology (NIST), 2001.

8. Standaert, F.-X., Örs, S.B., Quisquater, J.-J., Preneel, B.: Power analysis attacks against FPGA implementations of the DES In: Proceedings of the FPL 2004, 2004.

9. Benoit Badrignans, Jean Luc Danger, Viktor Fischer, Guy Gogniat, Lionel Torres: Security Trends for FPGAS: From Secured to Secure Reconfigurable Systems, June 20, 2011.

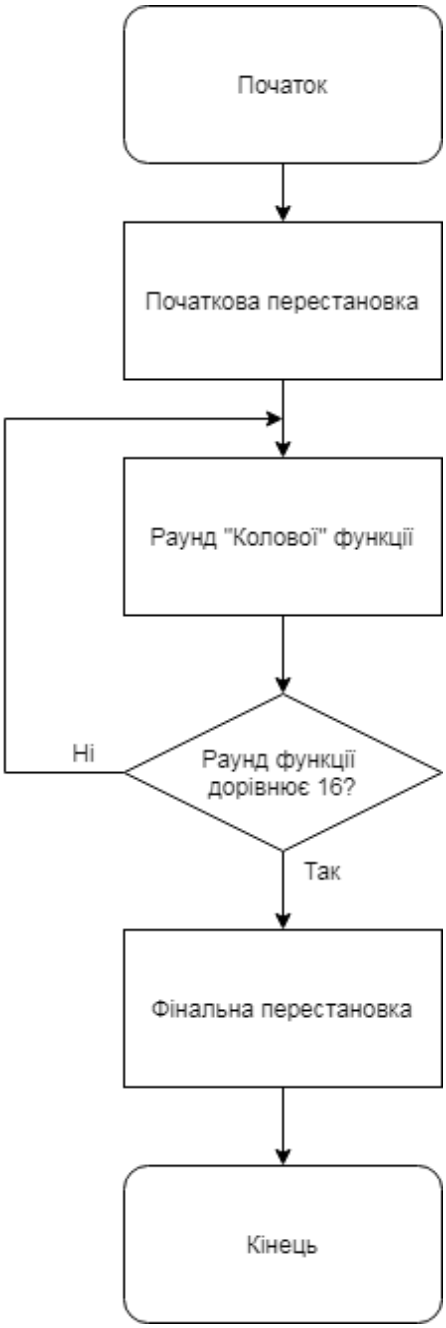
10. Vikram Pasham and Steve Trimberger: High-Speed DES and Triple DES Encryptor/Decryptor, August 03, 2001.

11. H. Eberle. A high-speed DES implementation for network applications. Technical Report 90, Digital Equipment Corporation Systems Research Center, September, 1992.

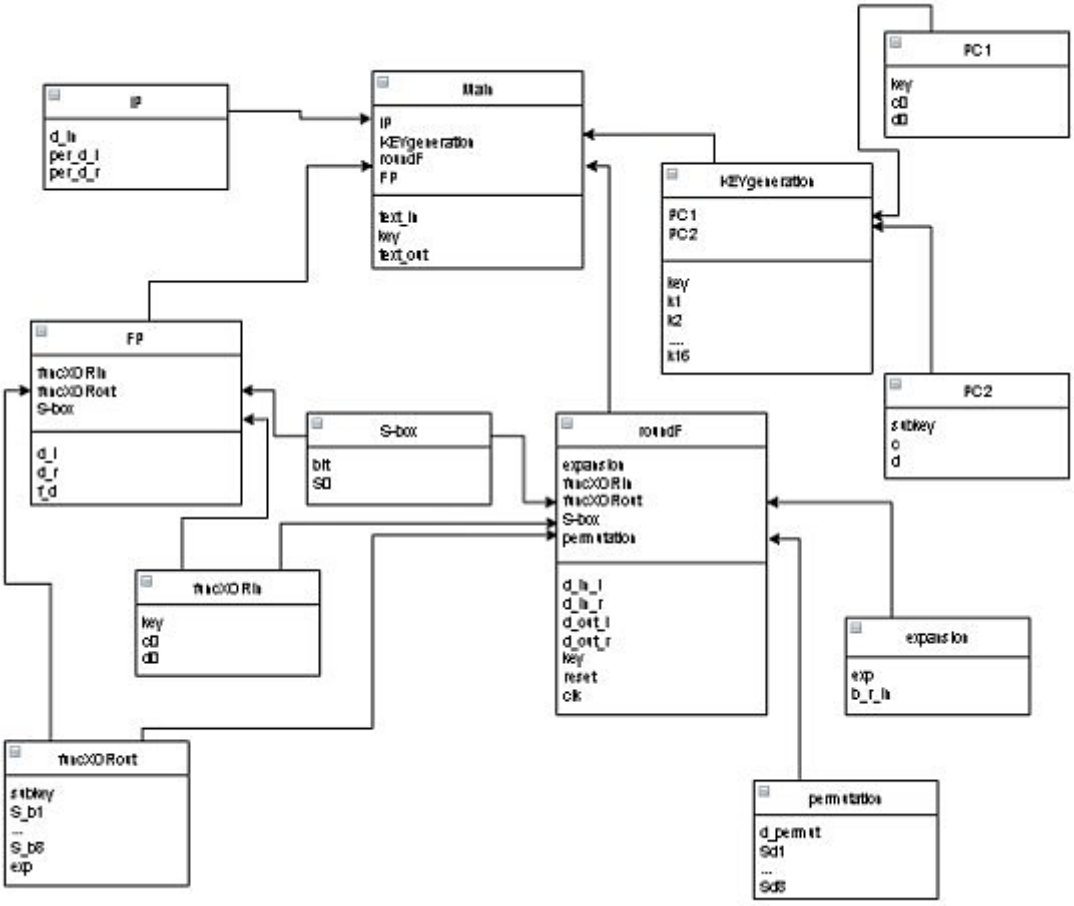
12. Cryptography, Theory and Practice by Stinson, Nov, 2005.

13. Biclique Cryptanalysis of the Full AES, March, 2016.

14. Daemen, Joan; Rijmen, Vincent: AES Proposal: Rijndael. National Institute of Standards and Technology, March, 2013.



					ІАЛЦ.467200.004 Д1	
Изм.	Лист	ПІБ	Підпис	Дата		
Розроб.		Гудзенко О.Ю			Блок-схема алгоритму DES. Додаток 1.	Аркуш 1 / Аркушів 1
Керівник		Сергієнко А.М.				КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62
Консультант						
Н/контр.		Сімоненко В.П.				
Зав. каф.						



					ІАЛЦ.467200.005 Д2		
Изм.	Лист	ПІБ	Підпис	Дата			
Розроб.		Гудзенко О.Ю			Принципова схема програми. Додаток 2.	Аркуш	Аркушів
Керівник		Сергієнко А.М.				1	1
Консультант						КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62	
Н/контр.		Сімоненко В.П.					
Зав. каф.							

					ІАЛЦ.467200.006 Д3	
Изм.	Лист	ПІБ	Підпис	Дата		
Розроб.		Гудзенко О.Ю			Структурна схема програми. Додаток 3.	Аркуш 1 / Аркушів 1
Керівник		Сергієнко А.М.				
Консультант						КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62
Н/контр.		Сімоненко В.П.				
Зав. каф.						